A Bridge-based Compression Algorithm for Topological Quantum Circuits

Wei-Hsiang Tseng, Chen-Hao Hsu, Wan-Hsuan Lin, and Yao-Wen Chang, *Fellow, IEEE*

***Abstract*—Topological quantum error correction (TQEC) is promising for scalable fault-tolerant quantum computation. The required resource of a TQEC circuit can be modeled as its space-time volume of a three-dimensional geometric description. Implementing a quantum algorithm with a reasonable physical qubit number and computation time is challenging for large-scale complex problems. Therefore, it is desirable to minimize the space-time volume for large-scale TQEC circuits. Previous work proposed bridge compression, which can significantly compress a TQEC circuit, but it was performed manually. This paper presents the first automated tool that can perform bridge compression on a large-scale TQEC circuit. Our proposed algorithm applies the bridge compression technique to compactify TQEC circuits with modularization. Besides, we offer a time-ordering-aware 2.5D placement for compacting TQEC circuits and satisfying time-ordered measurement constraints. On the other hand, we suggest friend net-aware routing to effectively reduce the required routing resource under topological deformation. Compared with the state-of-the-art work, experimental results show that our proposed algorithm can averagely reduce space-time volumes by 84%.**

***Index Terms*—Physical Design, Topological Quantum Error Correction, Bridge Compression, Quantum Design Automation, Quantum Computing, Braided Quantum Circuit, Space-time Volume Minimization.**

I. Introduction

Quantum computing has attracted much attention in recent years due to its capabilities in achieving substantial speedup on several classes of problems (e.g., factorization [1] and unstructured database search [2]) that are considered intractable in classical computing. However, large-scale quantum computing is challenging because quantum devices could suffer from significant noise from the environment and may thus produce faulty results. Therefore, fault-tolerant quantum circuits are needed for the scalability and reliability of quantum computing.

The preliminary version of this paper was presented at the 2021 ACM/IEEE Design Automation Conference (DAC'21), Las Vegas, NV, June 2021

This work was partially supported by AnaGlobe, ASE, Maxeda, Synopsys, TSMC, MOST of Taiwan under Grant MOST 108-2221-E-002-097-MY3, MOST 108-2911-I-002-544, and MOST 109-2224-E-002-009.

W.-H. Tseng is with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan; email: whtseng@eda.ee.ntu.edu.tw

C.-H. Hsu is with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan; email: chhsu@eda.ee.ntu.edu.tw

W.-H. Lin is with the Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan; emails: b06901054@ntu.edu.tw

Y.-W. Chang is with the Department of Electrical Engineering and Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan; email: ywchang@ntu.edu.tw

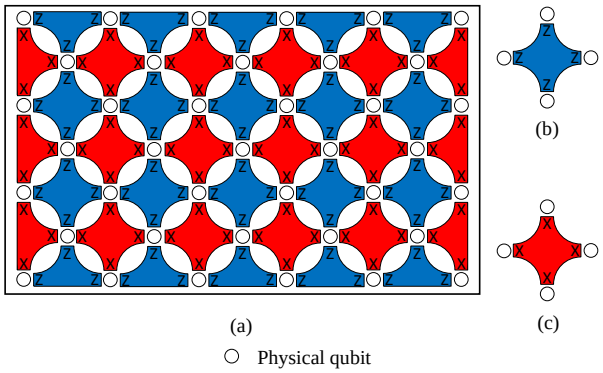

Fig. 1. (a) Surface code. (b) Z stabilizer. (c) X stabilizer.

The topological quantum error correction (TQEC) scheme is promising for scalable fault-tolerant quantum computation [3], [4]. Based on the Raussendorf code [5], quantum information is encoded into topological cluster states in a three-dimensional (3D) lattice structure [6] consisting of physical qubits. In the surface code [3], physical qubits are placed in a two-dimensional (2D) surface, as shown in Figure 1(a). Error correction of the quantum system is performed by periodically measuring four-qubit operators, known as *stabilizers*. Figure 1(b) and Figure 1(c) show a Z stabilizer and an X stabilizer respectively. Quantum computation in the surface code is achieved by manipulating so-called *defects*, specific contiguous regions where the stabilizers are turned off in the lattice. According to the type of stabilizers that are turned off, a defect is either primal (X) or dual (Z).

Most encoded logical operations in the surface can be performed by moving defects around each other, and defect movement is achieved by turning off the stabilizers in the new regions and then turning on the stabilizers in the old regions. Such movements in a 2D space can be divided into time slices. By stacking these 2D time slices according to their timing, the 3D diagram of defects can represent a TQEC circuit. *Geometric description* is a 3D visual representation in the surface code, which describes the qubit initialization/measurement (I/M), the defect configuration, and the positions of state injections and state distillation boxes [7]. There are two kinds of bases: X-basis and Z-basis. The basis vectors used for X-basis (Z-basis) are $|+\rangle$ and $|-\rangle$ ($|0\rangle$ and $|1\rangle$). In a geometric description, we use geometric components to represent qubit I/M [7], as shown in Figure 2. We use red cuboids and blue cuboids to represent primal defects and dual defects respectively in a 3D space. Figures 2(a) and (b) represent

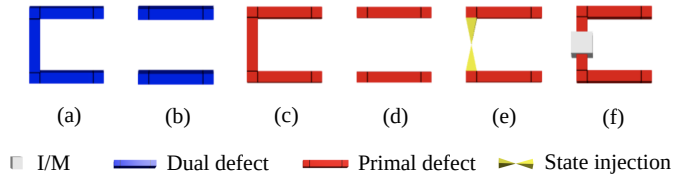

Fig. 2. The components used in geometric descriptions. (a) X-basis I/M for dual defects. (b) Z-basis I/M for dual defects. (c) Z-basis I/M for primal defects. (d) X-basis I/M for primal defects. (e) State injection to primal defects. (f) Generalized representation for initialization, measurement, and state injection.

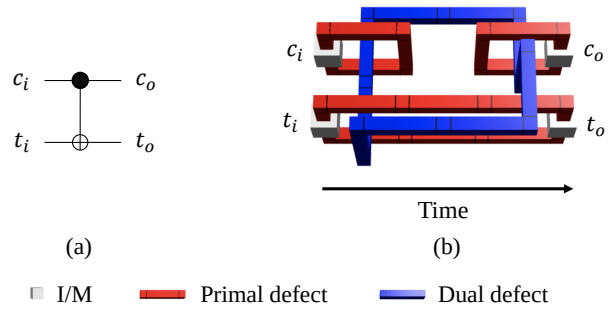

Fig. 3. (a) A CNOT gate. (b) A logical primal-primal CNOT gate.

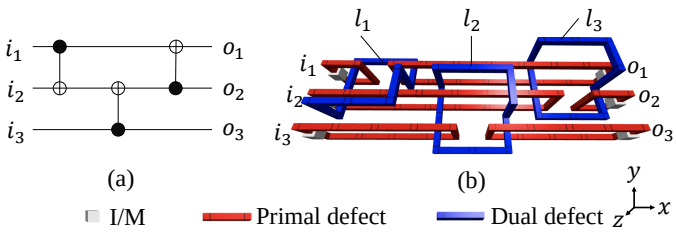

Fig. 4. An example of a quantum circuit with three CNOT gates. (a) The ICM representation. (b) The canonical geometric description of (a).

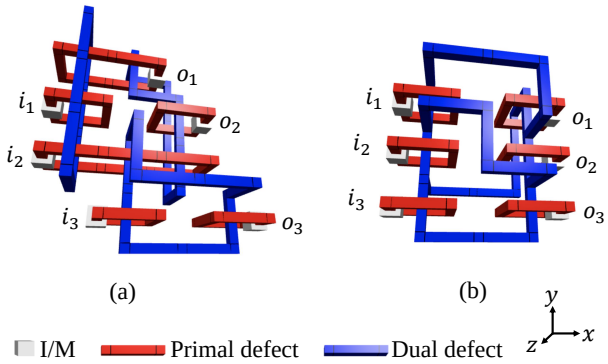

Fig. 5. Examples of our motivation. (a) The compressed geometric description after only topological deformation. (b) The optimized geometric description after bridge compression and topological deformation.

X-basis and Z-basis I/M for dual defects respectively. Figures 2(c) and (d) represent Z-basis and X-basis I/M for primal defects respectively. Figure 2(e) represents state injections for $|Y\rangle$ or $|A\rangle$ initialization. For generalization, a white cube represents the operations of initialization, measurement, and state injection, as shown in Figure 2(f).

In the surface code, a logical qubit is formed by a pair of same-type defects. To implement a logical primal-primal controlled-NOT (CNOT) gate whose logical input and output qubits are encoded in primal defects, we can braid the ancillary dual defects around the primal defects [7], as shown in Figure 3.

Furthermore, the error rate of a TQEC circuit is highly related to the distance between defects. Two defects of different types cannot overlap with each other, and two disjoint defect structures of the same type cannot overlap with each other as well. For simplicity, two defects are separated by one unit if they are not allowed to overlap with each other in this thesis.

We can convert any fault-tolerant quantum circuit into the ICM representation [8] that consists of qubit (I)nitialization, (C)NOT gates, and (M)easurements. Then, an ICM circuit can be easily mapped to a canonical geometric description [9]. For example, Figure 4(a) shows an ICM circuit with three CNOT gates, and Figure 4(b) shows the corresponding canonical geometric description with three dual loops l_1, l_2, and l_3 braided around primal loops. The functionality of a TQEC circuit remains unchanged under any topological deformation, which means that the deformed braids are topologically equivalent to the canonical braids [7].

The required resource of a TQEC circuit is abstracted to its space-time volume of the geometric description. The space volume represents the 2D quantum hardware resources (i.e., the number of physical qubits) for quantum computing, and the time volume represents the required executing time steps for quantum operations. To solve difficult problems in the real world, minimizing the space-time volume for large-scale TQEC circuits becomes crucial.

To lower the overhead of large-scale quantum computing, it is desirable to develop an automated tool to efficiently and effectively optimize the space-time volume of TQEC circuits. To the best of our knowledge, many studies focus on the optimization using topological deformation, but few works apply non-topological deformation such as bridge compression for automated space-time volume minimization, which can potentially compress TQEC circuits much more than the topological one.

Figure 4(b) shows a canonical geometric description with a volume of 54 (9×3×2). Figure 5(a) shows a compressed circuit with a volume of 32 (4×4×2) after only topological deformation, and Figure 5(b) shows an optimized circuit with a volume of 18 (3×3×2) after bridge compression and topological deformation. In this example, we can observe that bridge compression has great potential to achieve much lower space-time volume than only performing topological deformation. Note that the distance between two disjoint defects is one unit, and the volume is calculated by $(\#x \times \#y \times \#z)$ in this paper, where $\#x$, $\#y$, and $\#z$ are the numbers of units on the x, y, and z axes, respectively.

During the past decade, many approaches to synthesize and optimize TQEC circuits have been proposed. A survey of synthesis of TQEC circuit from an arbitrary quantum circuit is given in Section I-A. Also, surveys of depth optimization for

the ICM representation and space-time volume optimization for TQEC circuits are given in Section I-B and Section I-C respectively.

A. Synthesis of TQEC circuits

To realize a quantum algorithm, we first need to synthesize a sequence of quantum gates (operations) and map the gates to real hardware with the TQEC scheme. Therefore, several works focus on synthesizing correct and simple TQEC circuits in the surface code from an arbitrary quantum circuit [10], [11], [12], [13], [7], [14], [15].

Paler *et al.* [7] presented the first automated framework to synthesize a TQEC circuit from an arbitrary quantum algorithm. Given a quantum circuit described by a list of gates and the qubits it operates on, they first decomposed all the gates to TQEC supported gates. Then, the decomposed circuit was transformed into the ICM representation. Next, the ICM circuit can be directly mapped to a 3D canonical TQEC geometric description. Moreover, they also proposed a distillation scheduler for placing distillation boxes, where the introduction to distillation boxes will be given in Section II-A.

Paler *et al.* [14] developed a systematic and online method for synthesizing compact TQEC circuits. Once an ancillary state (i.e., $|Y\rangle$ or $|A\rangle$) is successfully produced with a high fidelity, it should be connected to the circuit initialization by defect segments. Thus, they proposed an algorithm for connecting the output of a box to the destination initialization through an area called *connection pool*. Furthermore, they proposed an online box scheduler to deal with possible distillation failures. Finally, they also proposed a spiral and layered structure for distillation box placement to achieve a lower space-time volume.

However, both of the works [7], [14] focus on synthesizing general and correct TQEC circuits and do not contain effective volume optimization techniques. Therefore, both of the synthesis methods may not be able to generate TQEC circuits for large-scale practical problems with reasonable computation resources.

B. Depth Optimization for ICM Representation

Because ICM representation is favorable for topological quantum computation [8], minimizing the number of lines or time steps (i.e., depth) in ICM models can potentially reduce the space-time volume of the corresponding TQEC circuits. Therefore, minimizing the number of lines and the depth of an ICM circuit has attracted much attention recently.

AlFailakawi *et al.* [16] developed a hybrid approach combining a left-edge greedy heuristic with a genetic algorithm (GA) to minimize the depth of topological quantum circuits in the ICM representation. They first used GA to find a good qubit ordering, and then the left-edge heuristic was performed to reduce the depth of the ICM circuit based on the qubit ordering. Moreover, if multi-target CNOT gates are allowed, their proposed method can merge gates together without changing the circuit functionality, which can further reduce the circuit depth.

Paler and Wille [17] proposed a wire recycling algorithm to reduce the overhead of topological quantum circuits. They devised a directed acyclic graph (DAG) called *causal graph* to indicate the temporal ordering of initialization, gates, and measurements. Based on the causal graph of an ICM circuit, two optimization heuristics were proposed to recycle ordered wires and unordered ones.

Adnan and Yamashita [18] presented a 2D logical qubit arrangement algorithm for ICM representation in order to reduce the number of time steps. They first identified a set of gate groups by solving a clique cover problem, where each gate group (i.e., a clique) consists of possibly non-overlapped gates in the ICM representation. Then, for each gate group, they enumerated all possible permutations of logical qubit arrangements to find a valid arrangement that all gates in the group can be executed in one time step. They used πDD [19] to efficiently maintain the permutations of logical qubit arrangements. In most cases, their proposed method can find an optimal logical qubit arrangement, but it suffers from long runtime for large cases. Therefore, they also proposed a simulating annealing (SA)-based heuristic to significantly improve runtime with little solution quality loss.

C. Space-time Volume Optimization for TQEC Circuits

To implement a quantum circuit with reasonable computation resources in the TQEC scheme, it is necessary and desirable to reduce the space-time volume of the synthesized 3D geometric description. Therefore, several methods of the space-time volume optimization for TQEC circuits have been proposed in recent years.

Fowler and Devitt [20] presented a non-topological deformation called *bridge compression* to compactify a TQEC circuit with manual efforts substantially. They provided a step-by-step optimization process of a $|Y\rangle$ state distillation circuit and an $|A\rangle$ state distillation circuit, which shows the effectiveness of bridge compression. However, we cannot perform manual optimization on large TQEC circuits.

Paetznick and Fowler [21] presented two compression algorithms for TQEC circuits. First, they proposed an efficient force-directed algorithm that smoothly pushes or pulls defect segments in a greedy way without destroying the braiding relationship. Nevertheless, the forced-directed algorithm may be stuck in a local minimum, and the solution quality is highly related to the initial canonical geometric description. Second, they proposed an SA-based algorithm. The entire braided circuit can be described by a set of cuboids in a mathematical form, where the cuboids must satisfy a set of constraints. Then, the SA process is applied to explore more solutions by accepting or rejecting the neighboring solution of the current solution. A better neighboring solution would always be accepted, while a worse neighboring solution would be accepted with a possibility, which can avoid local minima.

Lin *et al.* [22] proposed an efficient depth minimization algorithm for TQEC circuits with one-dimensional (1D) and 2D qubit arrangements, which selected non-conflict dual-defect routing patterns by solving a maximum weighted independent set problem. However, their approach only considered the

depth compression (along the time axis), so the space-time volume of a TQEC circuit may not be globally minimized. Furthermore, they proved the NP-hardness of the qubit routing problem in layout synthesis of TQEC circuits.

D. Lattice Surgery Technique

Many studies used the lattice surgery technique in surface code recently. Fowler and Gidney [23] compared the braiding technique to the lattice surgery technique by an algorithm with 108 T gates and 100 logical qubits. Ali *et al.* [24] explored different overheads when using planar (lattice surgery) and double-defect (braiding) encodings. These studies show that the lattice surgery technique costs less when the size is small due to the smaller size of planar lattices. However, the braiding technique becomes more efficient due to the better efficiency of braids compared to the slower swaps as the size increases.

On the other hand, Herr *et al.* [25] showed that lattice surgery optimization is NP-hard. Although the lattice surgery technique has potential advantages, there is still not much work effectively converting lattice surgery optimization into traditional EDA problems. Nevertheless, we can apply modularization [26] to model the TQEC compression problem as a placement-and-routing problem with the braiding technique. With the conversion, we could obtain the desired compression results to handle large-scale circuits. As a result, we focus on the compression of braiding topological quantum circuits in this paper.

We summarize our main contributions as follows:

- We present the first work that automatically optimizes the space-time volume of TQEC circuits by bridge compression while considering state distillation boxes and time-ordered measurement constraints.
- We develop an efficient iterative bridging algorithm to construct bridge structures, unlike the previous work that performs bridge compression with manual efforts.
- We propose time-ordering-aware 2.5D placement for compaction of TQEC circuits and the satisfaction of time-ordered measurement constraints.
- We propose an algorithm to merge primal modules into a primal-group super-module, which can efficiently reduce the problem size of the SA engine for 2.5D placement.
- We propose friend net-aware routing to reduce the required routing resource under topological deformation effectively.
- We offer the width, height, depth (time), and volume for experimental results, which show that our proposed algorithm can averagely achieve 83% space-time volume reduction compared with the state-of-the-art method [22].
- Experimental results show that integrating our proposed iterative bridging algorithm into TQEC optimization flow can averagely achieve a 28% volume reduction and an average 35% runtime speedup.
- We analyze the runtime breakdown of our proposed algorithm and conclude that a good module placement with less wirelength can facilitate the routing process. If we put much more effort into the placement stage, 85%–95% nets can be successfully routed in the first routing iteration.

The remainder of this paper is organized as follows. Section II introduces state distillation boxes, time-ordered measurement constraints, modularization, and the bridging rule, and then formulates the TQEC circuit compression problem. Section III details the core techniques of our algorithm. Section IV shows the experimental results, and Section V concludes this paper.

II. Preliminaries

This section briefly introduces state distillation boxes in TQEC circuits, time-ordered measurement constraints, the concept of modularization, and the bridge compression in Section II-A, Section II-B, Section II-C, and Section II-D, respectively. Finally, we formulate the TQEC circuit compression problem in Section II-E.

A. State Distillation Box

In the TQEC scheme, a single-qubit rotation gate can be implemented through teleportation with CNOT gates and logical ancillary states $|Y\rangle$ and $|A\rangle$ [7]:

$$|Y\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle), \tag{1}$$

$$|A\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\frac{\pi}{4}}|1\rangle). \tag{2}$$

These ancillary states must be prepared before injected into the circuit. State distillation circuits are used to generate a single high-fidelity ancillary state from multiple low-fidelity ones. Typically, we use a box to hold the place for a distillation circuit in geometric descriptions [7].

In this paper, we use the optimized state distillation circuits obtained manually in [20], where the volume of a $|Y\rangle$ box is 18 (3×3×2) and that of an $|A\rangle$ box is 192 (16×6×2), as shown in Figure 6 and Figure 7 respectively.

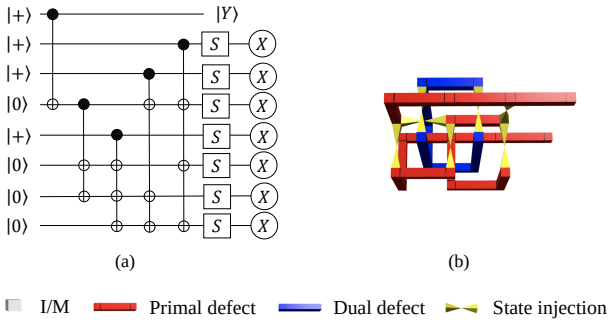

Fig. 6. The $|Y\rangle$ state distillation circuit. (a) The $|Y\rangle$ state circuit in the ICM representation. (b) The optimized TQEC circuit for $|Y\rangle$ state distillation.

B. Time-ordered Measurement Constraint

For TQEC circuits, most gates are invariant under any topological deformation. However, the measurements of certain gates should obey a relative time ordering in the ICM representation. For example, a T gate can be implemented through state injections along with gate teleportation [27]. Gate teleportation

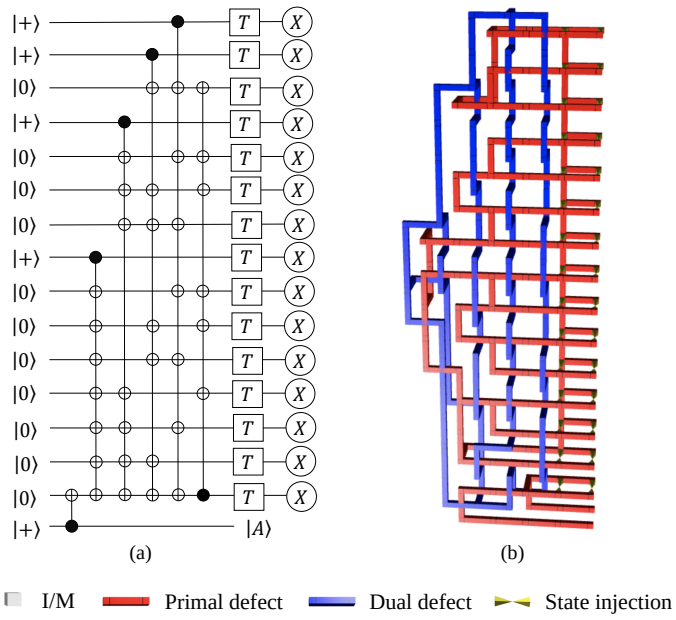

Fig. 7. The $|A\rangle$ state distillation circuit. (a) The $|A\rangle$ state circuit in the ICM representation. (b) The optimized TQEC circuit for $|A\rangle$ state distillation.

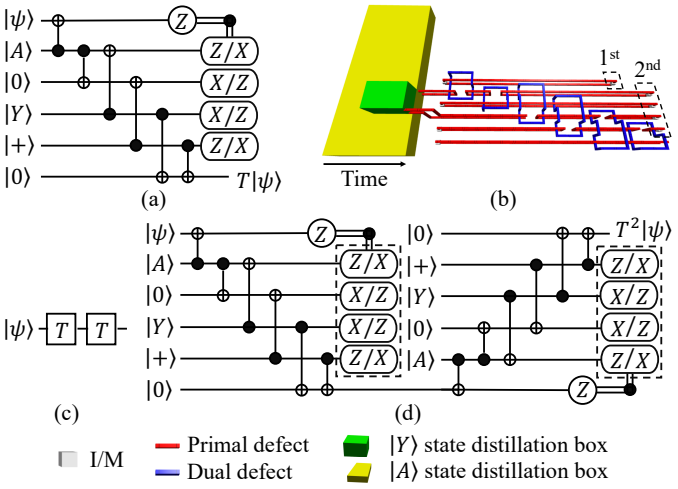

Fig. 8. (a) A T gate in the ICM representation with $|Y\rangle$ and $|A\rangle$ state injection . (b) The TQEC canonical form of a T gate with a $|Y\rangle$ box and an $|A\rangle$ box. (c) An example circuit with two T gates operating on the same qubit. (d) The ICM representation of (c).

contains probabilistic measurements, and the outcome of circuits depends on the measurement results [21]. Therefore, the input measurement should be performed before the selective teleportation measurements, which indicates the measurements of a T gate should obey a time ordering. Figure 8(a) shows a T gate in the ICM representation, where the topmost Z-basis measurement must be performed before the other four selective teleportation measurements [8]. Figure 8(b) shows a valid geometric description of the circuit in Figure 8(a), which meets the time-ordered measurement constraint. Note that in this thesis, time goes from left to right.

Furthermore, the selective teleportation measurements of distinct T gates operating on the same qubit in the ICM representation should also obey a time ordering [8]. The selective teleportation measurements of a T gate must be performed after those of the previous T gate are performed. Figure 8(c) shows a circuit with two T gates operating on a qubit, and the corresponding ICM representation is shown in Figure 8(d). To transform the circuit into a geometric description, we need to ensure that the selective teleportation measurements of the first T gate (the left dotted box) must be performed before the selective teleportation measurements of the second T gate (the right dotted box) are performed.

C. Modularization

Asai and Yamashita [26] proposed modularization that transforms the complicated TQEC circuit compression problem into a placement-and-routing problem. Modules of a TQEC circuit are derived from the canonical form by breaking all dual loops into a set of two-pin nets. The parts of dual loops penetrating a primal loop are kept in modules to preserve the braiding information, so a module consists of a primal loop enclosing dual segments.

Figure 9 shows an example of modularization of a quantum circuit with three CNOT gates. First, the ICM circuit in Figure 9(a) will be transformed into its canonical TQEC circuit shown in Figure 9(b). Then, by breaking all the three dual loops l_1, l_2, and l_3, there are six modules and nine dual-defect nets derived from the canonical form, as shown in Figure 9(c), where m_i denotes the i^{th} module. Finally, Figure 9(d) shows the labels of pins in each module, where $p^i_{j,k}$ denotes the k^{th} pin of the j^{th} dual segment enclosed by module m_i.

After the modularization of the canonical TQEC circuit, all the modules should be placed in the 3D space while the total volume, estimated total wirelength of all dual-defect nets, and routability should be optimized. Finally, the dual-defect nets should be routed to restore the dual loops. Note that the nets of a dual loop can be reconfigured as long as the dual loop can be restored. For example, both $\{(p^2_{3,2}, p^5_{1,2}), (p^5_{1,1}, p^4_{2,1}), (p^4_{2,2}, p^2_{3,1})\}$ and $\{(p^2_{3,2}, p^5_{1,2}), (p^5_{1,1}, p^4_{2,2}), (p^4_{2,1}, p^2_{3,1})\}$ are valid net sets for l_3 in Figure 9.

D. Bridge Compression

The bridge compression technique is proposed to lower the TQEC computation overhead by Fowler and Devitt [20]. A bridge can only be added between two disjoint finite extent same-type defect structures. After adding a bridge, the two structures are merged by a continuous common segment, defined as the segments of the two structures that pass through the same loops in the identical order. To achieve better circuit compaction, we should bridge two structures with a longer continuous common segment.

Figure 10(a) shows an initial TQEC circuit. To obtain the longest continuous common segments, we can topologically deform the TQEC circuit because the topological deformation maintains the braiding relationship between primal defects and dual defects, as shown in Figure 10(b). Two same braiding relationships can get identical computation results. Then, a

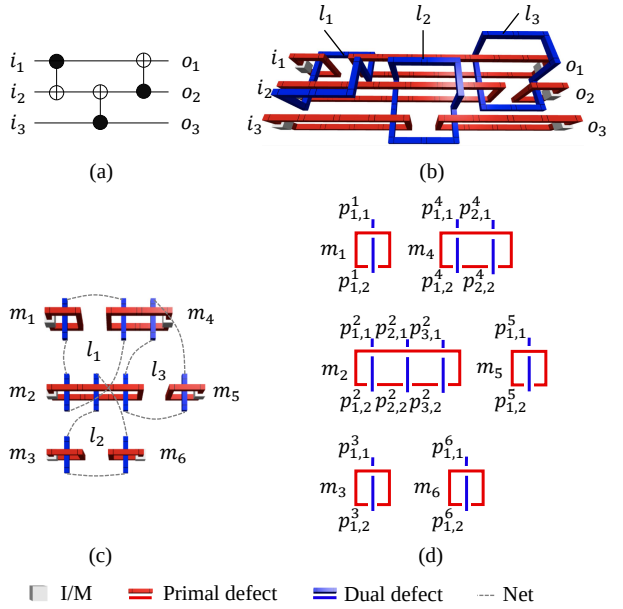

Fig. 9. An example of modularization. (a) An ICM circuit with three CNOT gates. (b) The canonical TQEC circuit of (a). (c) The modules and nets derived from (b). (d) The labels of pins in each module.

bridge is added between the two dual loops (structures) shown in Figure 10(c), and Figure 10(d) shows the resulting bridge structure consisting of two dual loops by merging the common segment.

Note that we can add only one bridge between two structures; that is, the two structures would be merged by only one continuous segment. Otherwise, an extra loop would be induced, which will change the computation and thus is forbidden. For example, Figure 10(e) shows a wrong bridge structure from Figure 10(a) because there are two bridges added between two dual loops. In Figure 10(f), an extra dual loop is thus induced in the middle by merging two common segments from Figure 10(e). An extra dual loop breaks the braiding relationship and causes errors in the computation results. Although we can bridge two disjoint primal/dual structures, however, we only add a bridge between dual structures to simplify and tackle the TQEC circuit compression problem in this thesis.

E. Problem Formulation

The TQEC circuit compression problem can be formally defined as follows:

- The TQEC Circuit Compression Problem: Given a quantum circuit synthesized from a quantum algorithm, generate a 3D geometric description for TQEC computation such that the space-time volume is minimized while the time-ordered measurement constraints are satisfied and state distillation boxes are integrated.

III. Proposed Algorithms

The space-time volume optimization of TQEC circuits is complicated due to the maintenance of braids, the integration

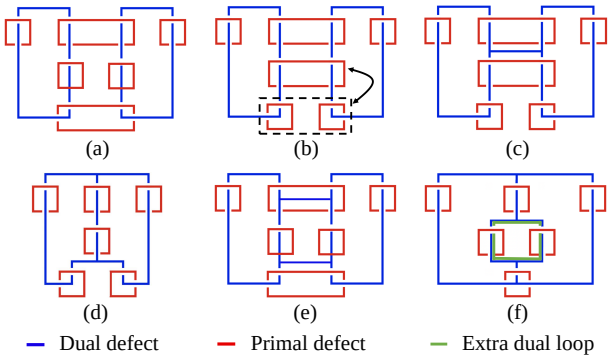

Fig. 10. (a) The initial TQEC circuit. (b) The circuit after topological deformation. (c) A bridge added between two dual loops from (b). (d) A dual bridge structure consisting of two dual loops derived by merging the common segment from (c). (e) Two bridges added between two dual loops from (a). (f) A wrong bridge structure from (e).

with state distillation boxes, and the time-ordered measurement constraints. Therefore, we propose an algorithm to optimize the space-time volume of TQEC circuits by bridge compression and topological deformation with the aid of modularization [26]. Our proposed algorithm consists of four major stages: (1) preprocess including gate decomposition and modularization, (2) iterative bridging, (3) module placement, and (4) dual-defect net routing. Figure 11 shows the overall flow of our proposed algorithm. The following subsections detail each stage.

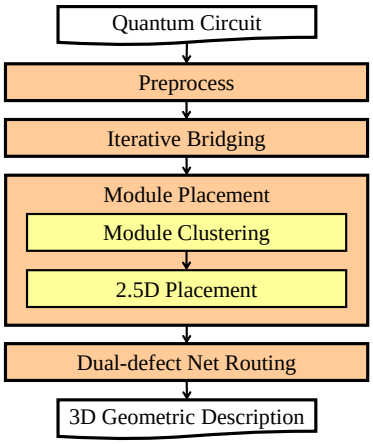

Fig. 11. Overview of our algorithm.

A. Preprocess

Each quantum computing architecture supports a specific gate set for universal computations. For TQEC computations, we apply the method [7] to decompose the gates of an arbitrary quantum circuit into the universal gate set $\{CNOT, P, V, T\}$ for the TQEC scheme, where the permutation matrix forms of each gate are listed as follows:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad (3)$$

$$P = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \tag{4}$$

$$V = \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}, \tag{5}$$

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}. \tag{6}$$

However, TQEC does not directly support the implementation of Toffoli gates that are widely used in classical reversible circuits. In order to implement a Toffoli gate, we need to decompose it into a sequence of CNOT, P, T, and H gates [28], as shown in Figure 12. Similarly, a Hadamard (H) gate can be decomposed into the P, V, P gate sequence [7]. Figure 13 shows the implementation of TQEC supported gates including P, H, and T gates.

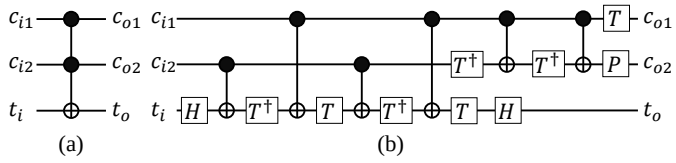

Fig. 12. (a) A Toffoli gate. (b) The decomposed Toffoli gate.

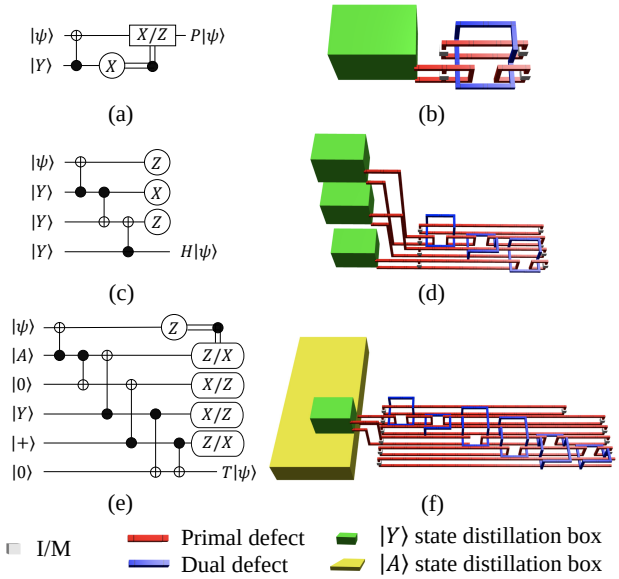

Fig. 13. The TQEC supported gates. (a) The P gate in the ICM representation. (b) The TQEC circuit for the P gate. (c) The H gate in the ICM representation. (d) The TQEC circuit for the H gate. (e) The T gate in the ICM representation. (f) The TQEC circuit for the T gate.

After gate decomposition, the decomposed quantum circuit with only TQEC supported gates can be transformed into the ICM representation. Next, the ICM circuit is mapped to the canonical geometric description, which only takes linear time with respect to the number of CNOT gates. According to the canonical form, a TQEC circuit is then decomposed into modules and dual-defect nets by modularization, as described in Section II-C.

B. Iterative Bridging

After the preprocessing stage, the iterative bridging is performed to merge dual loops into bridge structures by adding bridges. A bridge structure is composed of several bridged dual loops. For example, there is one bridge structure in Figure 10(d). During the bridging process, each dual loop maintains a set of *chains* for the flexibility of bridging. Each chain is a sequence of consecutive pins, and the starting pin and the ending pin are both called *endpoint pins*. Initially, for every loop, the two pins of each penetrated module form a chain. After a loop is merged to a bridge structure, the chains of loops in the bridge structure may be extended since they form a continuous common segment with the merged loop. Besides, a loop in a bridge structure is said to be *reconstructable* if its chains can be restored to a single loop by connecting all of its chains. Moreover, if two dual loops penetrate the same module, the module is called a *common module* of the two loops. For example, in Figure 9(d), module m_4 is a common module of l_1 and l_3 since both dual loops penetrate m_4. Furthermore, if loop l_i has at least one common module with loop l_j, then l_i is called a *relative loop* of l_j and vice versa.

Algorithm 1 shows our proposed iterative bridging algorithm to iteratively add bridges and merge dual loops into a bridge structure. First, all dual loops are pushed into set O and marked as unprocessed (line 1). Once a dual loop is merged to a bridge structure, it will be removed from O and marked as processed. In each iteration, we select an unprocessed loop l_i from O as the initial bridge structure b (line 4) and push the unprocessed relative loops of l_i into the max-priority queue Q (lines 5–6), which are candidate loops that could be merged to b. The priority of a loop in Q is determined by its number of common modules with b since we tend to merge a loop into a bridge structure with a potentially longer continuous common segment. Moreover, the loops in Q are extracted sequentially until Q is empty (lines 8–9), and we check whether the extracted loop l_e can be merged to b (lines 10–12). If l_e fails to be merged to b, it would never be pushed into Q in the current iteration. On the other hand, if l_e can be successfully merged to b (line 13), the chains of loops in b will be updated according to the continuous common segment of b and l_e (line 14). Then, l_e's unprocessed relative loops are pushed into Q as merged candidates (line 15). Besides, the keys to loops in Q would be updated accordingly since l_e is merged to b (line 16). Finally, l_e would be removed from O and marked as processed (line 17). This iterative bridging algorithm's worst-case and best-case time complexity are $O(n^2)$ and $O(n)$, respectively. Here n represents the number of dual loops.

Loop l_e could be merged to bridge structure b if we can find a continuous common segment that penetrates all common modules of b and l_e, and the continuous common segment cannot destroy the reconstructability of all loops in b. To find a continuous common segment of b and l_e, we construct a bridge graph $G_{b,l_e} = (V, E)$. The bridge graph construction consists of two steps: vertex construction and edge construction.

1) Vertex Construction: First, the pins of common modules between b and l_e are vertices in G_{b,l_e}. Second, if two chains

Algorithm 1 *IterativeBridging(D)*

Input: D: the set of dual loops.
Output: B: the set of bridge structures.
Push all dual loops in D into set O
$B \leftarrow \emptyset$
while O is not empty
  Select loop l_i from O as initial bridge structure b
  Initialize max-priority queue Q
  Push the unprocessed relative loops of l_i into Q
  Remove l_i from O and mark l_i processed
  while Q is not empty
    Extract loop l_e from Q
    Construct bridge graph G_{b,l_e}
    Determine the order of critical vertices
    Perform path finding for critical vertices
    if a valid path exists in G_{b,l_e}
      Merge l_e into b and update the chains of loops in b
      Push the unprocessed relative loops of l_e into Q
      Update the keys to loops in Q
      Remove l_e from O and mark l_e processed
  $B \leftarrow B \cup b$
return B

belonging to different loops in b share a common endpoint pin, then the endpoint pin is also a vertex in G_{b,l_e}. Note that for a bridge structure, we only need to consider one dual segment in a module. That is, each module contains only two pins for a bridge structure, so we use m'_i to denote a module instead of m_i in b. The vertex derived from pin p'^i_j is denoted by v^i_j, where p'^i_j denotes the j^{th} pin of module m'_i in b.

2) Edge Construction: First, if $p^{i_1}_{j_1}$ and $p^{i_2}_{j_2}$ are endpoints of different chains within a loop, then edge $e(v^{i_1}_{j_1}, v^{i_2}_{j_2})$ is constructed in G_{b,l_e}. Second, for each pair of consecutive pins $p^{i_3}_{j_3}$ and $p^{i_4}_{j_4}$ in a chain, then edge $e(v^{i_3}_{j_3}, v^{i_4}_{j_4})$ would be constructed if both $v^{i_3}_{j_3}$ and $v^{i_4}_{j_4}$ exist in G_{b,l_e}.

To merge l_e to b, the continuous common segment should penetrate all common modules of l_e and b. It implies that the vertices derived from the pins of common modules should be connected in series, and such vertices are called *critical vertices*. Therefore, our target is to find a path passing through all critical vertices in a specific order while the path does not destroy the reconstructability of each loop in b. Actually, the path indicates the continuous common segment for bridging b and l_e. Before path searching, we need to determine the connecting order of critical vertices. A valid connecting order can be obtained by following the two pin vertices of each common module sequentially. For example, for two common modules m'_1 and m'_2, the order $\langle v^1_1, v^1_2, v^2_1, v^2_2\rangle$ is valid, but the order $\langle v^1_1, v^2_1, v^2_2, v^1_2\rangle$ is invalid due to the discontinuity of the pin vertices of m'_1 (i.e., v^1_1 and v^1_2).

After the connecting order is determined, we perform path searching on G_{b,l_e} in order to find a path that follows the specific vertex order. Once a path is found in G_{b,l_e}, we will check if it is a valid path defined as a path that does not destroy the reconstructability of the loops in b. If a valid path is found, then l_e would be merged to b, and all the chains associated with the path would be updated. For edges in the valid path, if it connects two disjoint chains within a loop in b, the chains would be connected as one chain. The continuous common segment becomes a chain of l_e.

The whole iterative bridging process terminates when all dual loops are processed (i.e., O is empty). After all the bridge structures are constructed, we generate dual-defect nets by reconstructing all the loops in bridge structures, where all the nets should be connected in the following routing stage. Note that no duplicate nets will be generated.

Figures 14, 15, and 16 show an example of performing iterative bridging on the circuit in Figure 9. We want to know whether the dual defects in modules can be merged in Figure 9(d). Consequently, we need to iteratively add legal dual loops to construct bridge structures. First, l_1 is selected as the initial bridge structure b in Figure 14(a), and the chain set of l_1 is $\{p'^1_1$–p'^1_2, p'^2_1–p'^2_2, p'^4_1–$p'^4_2\}$ in Figure 14(b). Because l_2 and l_3 respectively share one common module (m'_2) and two common modules (m'_2 and m'_4) with l_1, they are pushed into the max-priority queue Q with keys 1 and 2, respectively. The key of a dual loop in the max-priority queue Q is the number of common modules with the current bridge structure b because we intend to bridge two structures with a longer common segment.

To check and process the merged candidate, we extract l_3 from Q in Figure 15. As shown in Figure 15(b), m'_2 and m'_4 are common modules of b and l_3 (i.e., l_1 and l_3). Then, we intend to construct the bridge graph G_{b,l_3} for b and l_3, as shown in Figure 15(c), to check if l_3 can be merged to b. For vertex construction, since m'_2 and m'_4 are common modules between b and l_3, v^2_1, v^2_2, v^4_1, and v^4_2 are constructed by p^2_1, p^2_2, p^4_1, and p^4_2, respectively. For edge construction, because v^2_1 and v^4_1 are vertices derived from endpoint pins of different chains within l_1, $e(v^2_1, v^4_1)$ is constructed. Similarly, $e(v^2_1, v^4_2)$, $e(v^2_2, v^4_1)$, and $e(v^2_2, v^4_2)$ are also constructed. In other words, we construct the edges between any endpoint pins of different chains together. Besides, $e(v^2_1, v^2_2)$ is constructed since v^2_1 and v^2_2 are two consecutive pins in a chain. Similarly, $e(v^4_1, v^4_2)$ is constructed. Suppose that the connecting order is $\langle v^2_2, v^2_1, v^4_2, v^4_1\rangle$, a valid path for continuous common segment is shown in Figure 15(c). We can see that after bridging b and l_3 with the continuous common segment, the reconstructability of the loops in b is not destroyed because both l_1 and l_3 can be restored to a single loop by connecting all chains in the chain set, as shown in Figure 15(d). Furthermore, after l_3 is merged to b, the chain set of l_1 is updated as $\{p'^4_1$–p'^4_2–p'^2_1–p'^2_2, p'^1_1–$p'^1_2\}$, and that of l_3 is $\{p'^4_1$–p'^4_2–p'^2_1–p'^2_2, p'^5_1–$p'^5_2\}$.

Repeating the above steps for merging l_2 to b, the bridge graph G_{b,l_2} is constructed, as shown in Figure 16(c). Note that v^4_2 is constructed because p'^4_2 is the common endpoint of the two chains belonging to l_1 and l_3. Finally, eight dual-defect nets are generated by reconstructing all the loops in b, as shown in Figure 16(d).

C. Module Placement

In the placement stage, we need to place all the modules in 3D space with the optimization of total wirelength and routability while considering time-ordered measurements and state distillation boxes for universal computations. In this work, we use the optimized distillation boxes obtained in [20], where the volume of a $|Y\rangle$ box is 18 (3×3×2) and that of

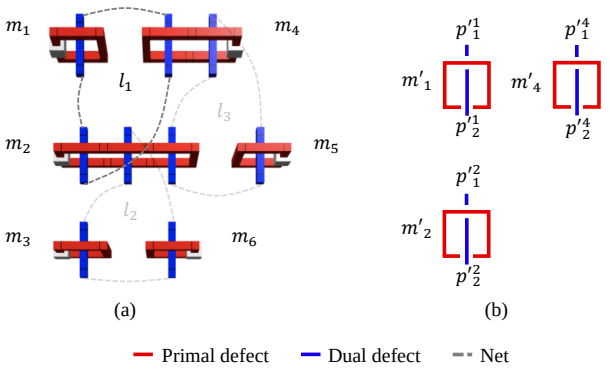

Fig. 14. An example of performing iterative bridging on the TQEC circuit in Figure 9. (a) l_1 is selected as the initial bridge structure b. (b) The initial bridge structure b.

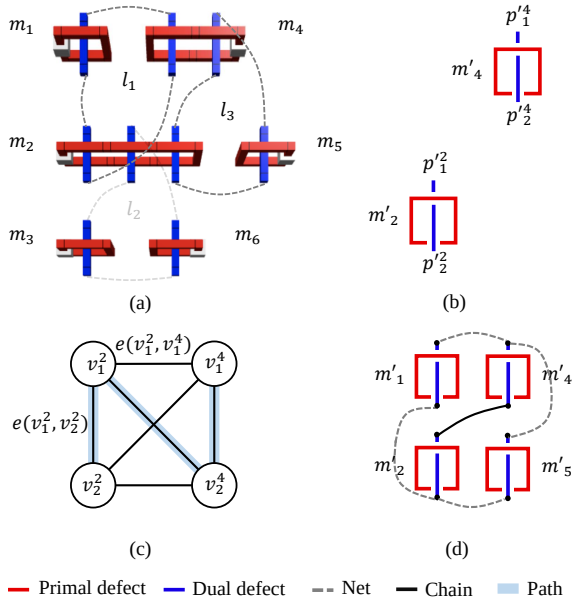

Fig. 15. An example of performing iterative bridging on the TQEC circuit in Figure 9. (a) l_3 is selected from the max-priority queue Q. (b) The common modules of b and l_3. (c) The bridge graph G_{b,l_3}. (d) The resulting bridge structure after adding l_3.

an $|A\rangle$ box is 192 (16×6×2). $|Y\rangle$ and $|A\rangle$ state distillation boxes are regarded as modules and should be placed as well. To satisfy the constraint or reduce the problem size, we cluster some primal modules into a *super-module* for state injections, time-ordered measurements, and primal-group module formation. Next, we perform 2.5D placement while considering wirelength, routability, and time-ordered measurement constraints. We use the 2.5D B*-tree representation [29] to model the modules (or super-modules), where each node denotes a module (or a super-module). The two major steps in the placement stage are detailed in this section. Section III-C1 introduces module clustering, and Section III-C2 details 2.5D module placement.

1) Module Clustering: Both time-ordered measurements and distillation boxes for state injections should be placed in certain ordering along the time axis. Therefore, we propose module clustering to handle the time-ordered measurement

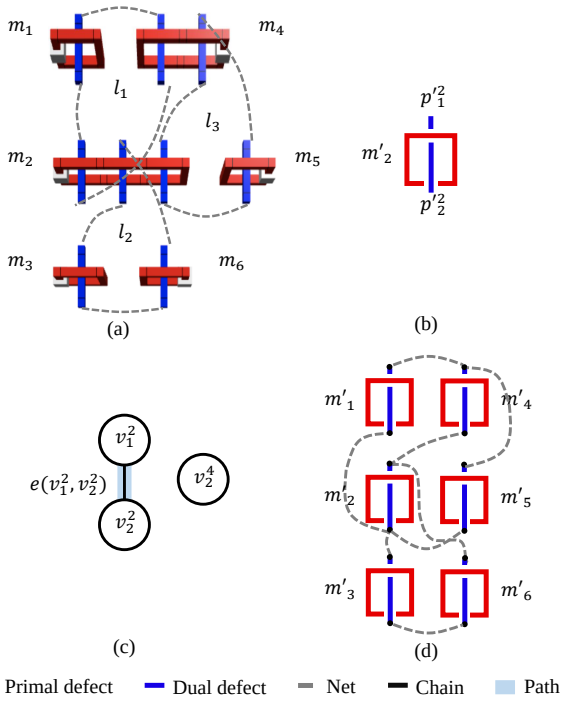

Fig. 16. An example of performing iterative bridging on the TQEC circuit in Figure 9. (a) l_2 is selected from the max-priority queue Q. (b) The common modules of b and l_2. (c) The bridge graph G_{b,l_2}. (d) The resulting bridge structure after adding l_2.

issue and the integration of state distillation boxes. There are three types of super-modules: time-dependent super-modules, distillation-injection super-modules, and primal-group super-modules. We detail each type of super-modules below.

- **Time-dependent super-module**: The T gate measurements in the ICM representation should be performed in a specific time ordering, as mentioned in Section II-B. Therefore, we cluster the modules associated with the five measurements of a T gate into a time-dependent super-module. To meet the time-ordering constraint of T gate measurements, the module associated with the first Z-basis measurement must be placed on the left side of the four modules associated with the four selective teleportation measurements of the T gate. Besides, the four modules are placed vertically and aligned by their right boundaries (i.e., measurements). By this arrangement, the first Z-basis measurement must be performed before the four selective teleportation measurements of a T gate. Figure 17(a) shows an example of the time-dependent super-module for a T gate, where the module on the left is associated with the first Z-basis measurement, and the four modules on the right are associated with the four selective teleportation measurements of the T gate. Note that the sizes of time-dependent super-modules can be different, depending on the sizes of modules associated with the measurements of a T gate.
- **Distillation-injection super-module**: Ancillary states $|Y\rangle$ or $|A\rangle$ should be prepared before injected into the circuit. Although a state distillation box can be placed

at any position ahead of the injected module in the time axis, we cluster a state distillation box and its injected module into a distillation-injection super-module by directly connecting them to reduce the required primal-defect routing resource between them and meet the distillation-injection constraint. Figure 17(b) shows a $|Y\rangle$ state distillation-injection super-module, and Figure 17(c) shows an $|A\rangle$ state distillation-injection super-module. Similar to time-dependent super-modules, the sizes of distillation-injection super-modules may be different, depending on the sizes of modules associated with the injection.

- **Primal-group super-module**: After the time-dependent super-module clustering and the distillation-injection super-module clustering, each of the remaining unclustered primal modules corresponds to a node in a 2.5D B*-tree, which could induce a considerable number of nodes and thus increase runtime. To reduce the number of nodes, we propose an efficient algorithm to cluster the primal modules directly connected by dual loops into a primal-group super-module. We examine the primal modules sequentially, and the primal modules connected by the same dual loop form a group until the upper limit of the number of groups is reached or no unprocessed primal modules remain. Since all the remaining primal modules are examined only once, the time complexity is linear to the number of the remaining primal modules. After the primal-group super-module clustering, the number of nodes in the 2.5D B*-tree would be significantly reduced.

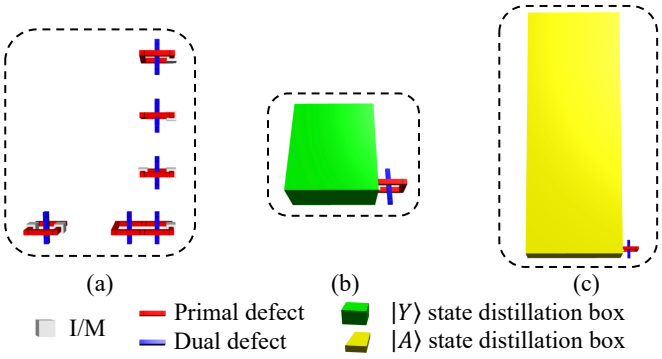

Fig. 17. Examples of super-modules. (a) A time-dependent super-module. (b) A $|Y\rangle$ state distillation-injection super-module. (b) An $|A\rangle$ state distillation-injection super-module.

2) 2.5D Placement: Instead of general 3D architectures, we propose to place all modules in a 2.5D architecture for its higher regularity, which can benefit the routability. For a 2.5D architecture, modules are divided into different tiers, where the modules in each tier are placed in a 2D plane. All tiers are stacked up to form a 2.5D architecture in the 3D space. We use the 2.5D B*-tree representation [29] for the modules (or super-module), and the placement of each tier is represented by a 2D B*-tree [30], where each node in the tree represents a module (or super-module). For example, Figure 18 shows an example of a three-tier 2.5D B*-tree representation and the corresponding B*-tree of each tier.

A simulated annealing (SA) engine is applied to minimize the total volume and total estimated wirelength. Similar to [29], we apply four perturbations of the 2.5D B*-tree to explore the solution space, including inter-/intra-tree node swap and inter-/intra-tree node move. Note that it is not allowed to rotate a module since the time ordering of the modules inside both time-dependent super-modules and distillation-injection super-modules would be changed after rotation. Furthermore, to enhance routability, each module is slightly expanded to preserve some routing regions around the module.

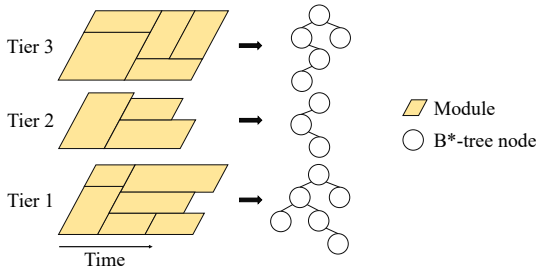

Fig. 18. An example of a 2.5D B*-tree representation with three tiers and the corresponding B*-tree for each tier.

To achieve a good trade-off between the total volume and the total estimated wirelength, we use the cost function Φ to evaluate the solution quality of a placement:

$$\Phi = \alpha \cdot \frac{V}{V_{norm}} + \beta \cdot \frac{L}{L_{norm}} + \gamma \cdot (R - R^*)^2, \qquad (7)$$

where V is the current placement volume, V_{norm} is the average placement volume, L is the current sum of total wirelength estimated by Manhattan distances of all nets, L_{norm} is the average sum of total wirelength, R is the current placement aspect ratio, R^* is the desired placement aspect ratio, and α, β, and γ are user-defined parameters. In our implementation, R^* is set to 1:2 (width:height), and α, β, and γ are set to 0.5, 0.5, 0.25, respectively.

The selective teleportation measurements among the T gates operating on the same qubit should obey a relative time ordering, as mentioned in Section II-B. To maintain the ordering during the SA process, we create a time-dependent super-module list (TSL) for each qubit. The time-dependent super-modules associated with the T gates operating on the same qubit are added to a TSL. Before the SA engine is applied, the super-modules in a TSL are resized to the same size. After a perturbation operation is performed on the 2.5D B*-tree, the positions of time-dependent super-modules would be identified. According to their relative time ordering, the time-dependent modules in a TSL will be reallocated to the identified positions. By the reallocation, the order of T gates in a TSL is maintained after a perturbation operation. Besides, the reallocation does not affect the positions of other modules since the super-modules in a TSL are resized to an identical size.

D. Dual-defect Net Routing

In this section, we detail our proposed dual-defect net routing algorithm. First, we introduce the routing framework

in Section III-D1. Next, Section III-D2 introduces the friend net awareness during routing.

1) Routing Framework: In the routing stage, we should route all the nets to restore the dual loops and bridge structures. We adopt an iterative routing scheme that all unrouted nets will attempt to be routed in each iteration once. In the first iteration, all the nets are routed sequentially in the non-decreasing order sorted by the Manhattan distance of each net, and the A* search algorithm is applied to route each net within a restricted search region. Initially, the search region of a net is the bounding box of its two pins.

After the first iteration, the rip-up and reroute scheme is applied to improve the routability for unrouted nets. The negotiation-based rip-up and reroute technique [31] is adopted to effectively alleviate routing congestion by maintaining a history map, which is a common technique widely used in electronic design automation (EDA) [32], [33], [34]. Moreover, if a net fails to be routed, the search region of the net will be slightly expanded in the next iteration to explore more routable regions.

The primal- and dual-defect segments in each module and state distillation boxes should be regarded as obstacles for all nets. Also, the routed nets should be viewed as obstacles for other nets, except for friend nets, which will be introduced later. Any net cannot overlap with obstacles during routing. Furthermore, we use an R-tree [35] to efficiently maintain all obstacles in a 3D space, where the average complexity of performing a search on an R-tree with n objects is $O(\lg n)$.

2) Friend Net Awareness: In this subsection, we introduce the concept of the *friend net*. If two nets share the same pin, then they are defined as a friend net to each other with respect to the pin. Once net n_i is routed, the friend nets of n_i with respect to one of n_i's pin p can end on any point of the routed path of n_i instead of p. This rule is a kind of topological deformation that does not change the braiding relationship and thus is valid.

Figure 19(a) shows an example that n_1 and n_2 are friend nets with respect to pin p. Therefore, after n_1 is routed, as shown in Figure 19(b), the unrouted net n_2 can end on any point of the routed path of n_1 instead of p, as shown in Figure 19(c). The friend net-aware routing can indeed reduce the required routing resource and also enhance the routability.

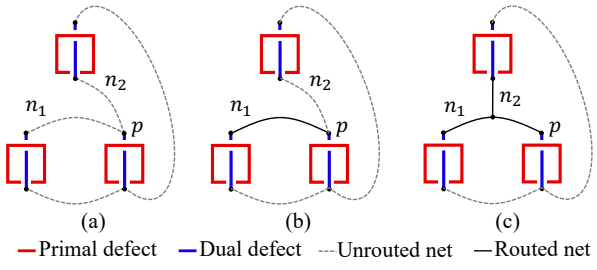

Fig. 19. The concept of the friend net. (a) All nets are unrouted. (b) n_1 is routed. (c) n_2 can end on any point of n_1 because they share the pin.

IV. Experimental Results

In this section, we show the experimental results of our proposed algorithm for the TQEC circuit compression problem. Section IV-A introduces the experimental setup. Next, Section IV-B gives the benchmark statistics. Finally, Section IV-C shows the experimental results of the proposed algorithm compared to the state-of-the-art method.

A. Experimental Setup

We implemented our algorithm in the C++ programming language with the Lemon graph library [37] and the Boost C++ libraries [38]. All experiments were performed on a Linux workstation with 4 Xeon 3.4 GHz CPUs with 64 GB memory.

B. Benchmark Statistics

The experiments were conducted on the RevLib benchmarks [39] that are widely used reversible circuits in quantum computing research. The benchmark statistics is summarized in Table I. First, "#Qubits$_o$" and "#Gates" denote the numbers of qubits and gates (including CNOT gates, Toffoli gates, etc.) respectively before the gate decomposition. Second, "#Qubits$_d$", "#CNOTs", "#$|Y\rangle$", and "#$|A\rangle$" denote the numbers of qubits (including input qubits and ancillas), CNOT gates, $|Y\rangle$ ancillas, and $|A\rangle$ ancillas, respectively, after the gate decomposition introduced in Section II. Besides, "Vol$_{|Y\rangle}$" and "Vol$_{|A\rangle}$" denote the total volume of $|Y\rangle$ state distillation boxes and that of $|A\rangle$ state distillation boxes respectively. Then, "#Modules" and "#Nets" denote the numbers of modules and nets respectively after modularization and iterative bridging. Further, "#Nodes" denotes the number of nodes in the 2.5D B*-tree after module clustering (the super-modules). Note that we can measure the problem size reduction of the SA algorithm by "#Modules" minus "#Nodes".

C. Results and Comparisons

In this subsection, we first compare our proposed algorithm with the state-of-the-art method [22] and our conference version [36] to show the effectiveness of the proposed algorithm in Section IV-C1. Next, we show the effectiveness of bridging in Section IV-C2 and the runtime breakdown of the proposed algorithm on each benchmark in Section IV-C3. Finally, the layouts of optimized TQEC circuits are shown in Section IV-C4.

1) Comparison with the State-of-the-art Method: The space-time volume comparisons of our proposed algorithm and the state-of-the-art method [22] and the conference version [36] are summarized in Tables II and III, respectively. Table IV gives the dimensions of the resulting TQEC circuits, where "W", "H", "D", and "Vol" denote the width, the height, the depth (time), and the volume of a TQEC circuit, respectively.

The total volumes (denoted by "Vol$_t$") of them shown in Table II and Table III are calculated by the original volumes (denoted by "Vol$_o$") of the synthesized TQEC circuits directly plus the total volume of state distillation boxes, which is the lower-bound volume, for a fair comparison. From Table II, we

TABLE I
BENCHMARK STATISTICS.

Benchmark	#Qubits$_o$	#Gates	#Qubits$_d$	#CNOTs	#$\|Y\rangle$	#$\|A\rangle$	Vol$_{\|Y\rangle}$	Vol$_{\|A\rangle}$	#Modules	#Nets	#Nodes
4gt10-v1_81	5	6	131	168	42	21	756	4032	362	483	190
4gt4-v0_73	5	17	257	341	84	42	1512	8064	724	978	384
rd84_142	15	28	897	1162	294	147	5292	28224	2500	3339	1316
hwb5_53	5	55	1307	1729	434	217	7812	41664	3687	4982	1933
add16_174	49	64	1394	1792	448	224	8064	43008	3857	5167	2032
sym6_145	7	36	1519	1980	504	252	9072	48384	4255	5688	2257
cycle17_3_112	20	48	1911	2478	630	315	11340	60480	5321	7119	2833
ham15_107	15	132	3753	4938	1246	623	22428	119616	10560	14215	5566

TABLE II
COMPARISON OF THE SPACE-TIME VOLUME FOR THE STATE-OF-THE-ART WORK [22] AND OURS.

Benchmark	Canonical		[22] (1D)			[22] (2D)			Ours		
	Volume	Ratio	Volume	Ratio	Runtime (s)	Volume	Ratio	Runtime (s)	Volume	Ratio	Runtime (s)
4gt10-v1_81	136836	5.509	98322	3.958	0.9	91116	3.668	0.8	24840	1.000	14
4gt4-v0_73	535398	9.222	361152	6.221	0.3	327816	5.647	0.3	58056	1.000	25
rd84_142	6287400	13.944	2805246	6.221	8	2744316	6.806	9	450912	1.000	194
hwb5_53	13608294	11.493	9114828	7.698	28	8203548	6.928	24	1184040	1.000	438
add16_174	15028608	15.667	6449532	6.723	26	6173928	6.436	23	959262	1.000	629
sym6_145	18103176	10.462	1072836	6.196	39	9852336	5.694	34	1730352	1.000	791
cycle17_3_112	28469700	15.455	19082448	10.359	71	16843884	9.144	61	1842050	1.000	1375
ham15_107	111335928	17.058	69294822	10.617	459	63017484	9.655	396	6527070	1.000	4108
Avg. Ratio		12.351		7.249			6.657			1.000	

TABLE III
COMPARISON OF THE SPACE-TIME VOLUME FOR THE CONFERENCE VERSION [36] AND OURS.

Benchmark	Conference version [36]			Ours		
	Volume	Ratio	Runtime (s)	Volume	Ratio	Runtime (s)
4gt10-v1_81	25520	1.027	15	24840	1.000	14
4gt4-v0_73	58696	1.011	26	58056	1.000	25
rd84_142	451440	1.011	262	450912	1.000	194
hwb5_53	1341704	1.133	447	1184040	1.000	438
add16_174	1069362	1.115	590	959262	1.000	629
sym6_145	1971840	1.140	793	1730352	1.000	791
cycle17_3_112	2354100	1.278	1402	1842050	1.000	1375
ham15_107	7331454	1.123	4901	6527070	1.000	4108
Avg. Ratio		1.104			1.000	

TABLE IV
THE DIMENSIONS AND TOTAL VOLUMES OF THE RESULTING TQEC CIRCUITS GENERATED BY THE STATE-OF-THE-ART WORK [22] AND OURS.

Benchmark	Canonical				[22] (1D)				[22] (2D)				Ours			
	W	H	D	Vol	W	H	D	Vol	W	H	D	Vol	W	H	D	Vol
4gt10-v1_81	131	2	504	132048	357	2	131	93534	327	8	33	86328	45	24	23	24840
4gt4-v0_73	257	2	1023	525822	684	2	257	351576	612	8	65	318240	59	41	24	58056
rd84_142	897	2	3486	6253884	1545	2	897	2771730	1506	8	225	2710800	122	112	33	450912
hwb5_53	1307	2	5187	13558818	3468	2	1307	9065352	3117	8	327	8154072	184	165	39	1184040
add16_174	1396	2	5376	15009792	2295	2	1394	6398460	2193	8	349	6122856	174	149	37	959262
sym6_145	1519	2	5940	18045720	3510	2	1519	10663380	3222	8	380	9794880	208	177	47	1730352
cycle17_3_112	1910	2	7434	28397880	4974	2	1911	19010628	4386	8	478	16772064	175	277	38	1842050
ham15_107	3753	2	14814	111193884	9213	2	3753	69152778	8370	8	939	62875440	330	347	57	6527070

TABLE V
COMPARISON OF THE SOLUTION QUALITY W/O AND W/ ITERATIVE BRIDGING.

Benchmark	W/o bridging				W/ bridging			
	Volume	Ratio	Runtime (s)	Ratio	Volume	Ratio	Runtime (s)	Ratio
4gt10-v1_81	33660	1.355	20	1.429	24840	1.00	14	1.00
4gt4-v0_73	76328	1.314	43	1.720	58056	1.00	25	1.00
rd84_142	640332	1.420	403	2.077	450912	1.00	194	1.00
hwb5_53	1659864	1.402	584	1.333	1184040	1.00	438	1.00
add16_174	1439064	1.500	740	1.176	959262	1.00	629	1.00
sym6_145	2509920	1.451	900	1.138	1730352	1.00	791	1.00
cycle17_3_112	2750895	1.493	1642	1.194	1842050	1.00	1375	1.00
ham15_107	8852480	1.356	6786	1.652	6527070	1.00	4108	1.00
Avg. Ratio		1.412		1.465		1.000		1.000

TABLE VI
RUNTIME BREAKDOWN.

Benchmark	Iterative bridging		Module placement		Dual-defect net routing		Other		Total (s)
	Time (s)	Ratio (%)	Time (s)	Ratio (%)	Time (s)	Ratio (%)	Time (s)	Ratio (%)	
4gt10-v1_81	0.11	0.77	13.10	91.6	1.08	7.55	0.01	0.07	14.30
4gt4-v0_73	0.25	1.01	20.92	84.76	3.47	14.06	0.04	0.07	24.68
rd84_142	2.15	1.11	151.37	78.18	39.84	20.57	0.25	0.13	193.61
hwb5_53	7.20	1.64	220.69	50.33	20.95	47.89	0.38	0.13	438.42
add16_174	6.77	1.07	234.31	37.26	386.78	61.51	0.99	0.16	628.85
sym6_145	8.58	1.08	536.09	67.73	245.69	31.06	0.64	0.08	791
cycle17_3_112	13.49	0.98	1043.32	75.89	317.15	23.07	0.83	0.06	1374.79
ham15_107	58.28	1.42	1999.18	48.67	2047.36	49.84	2.96	0.07	4107.78
Avg.	12.10	1.14	527.37	66.81	406.42	31.94	0.79	0.11	946.68

can see that the total volume of state distillation boxes only takes a little portion ($< 1\%$) of the total TQEC circuit volume in most benchmarks.

The work [22] proposes 1D and 2D qubit arrangements, and the results of both arrangements are also reported in Table II, where 2D qubit arrangements do result in less space-time volumes than 1D ones. Our proposed algorithm can averagely achieve 91.0% volume reduction from the canonical forms while the work [22] can only achieve 46.3% reduction. In other words, our proposed method can achieve 84% volume reduction from the work [22] with 2D qubit arrangements. From Table I, the conference version [36] incurs too many nodes in the 2.5D B*-tree representation for searching the desired result by the SA engine for large-scale benchmarks. We reduce the problem size by about half by constructing the primal-group super-modules. On the other hand, the modules that make up the primal-group super-modules are highly correlated (connected to the same dual loops), which helps get a better initial solution for the SA engine. After reducing the problem size, we can easily get a better initial solution by the SA engine and search for the desired result for a large-scale circuit. Overall, our proposed method can achieve a 9% volume reduction over the conference version [36]. The experimental results justify the effectiveness of our proposed compression algorithm.

2) Effectiveness of Bridging: To evaluate the effectiveness of our proposed iterative bridging algorithm, we further conducted experiments with and without the proposed iterative bridging method. Table V gives the compression results. With bridging, the volume for each benchmark is averagely reduced by 41%. Besides, the runtime is significantly reduced for each benchmark with bridging as well. To be more specific, the bridging contributes averagely 46% runtime speedup.

Both the TQEC circuit volume and the runtime for each benchmark can be reduced when the iterative bridging is applied mainly due to the following two reasons. First, our proposed iterative bridging can bridge many dual loops into bridge structures. Therefore, the number of dual-defect nets can be reduced, which diminishes the routing complexity. Second, because the proposed friend net-aware routing cannot be applied without bridging, the required routing resource thus increases, which causes larger space-time volume for TQEC circuits.

3) Runtime Breakdown: Table VI shows the runtime breakdown of our proposed algorithm on each benchmark. "Other" includes parsing input files, preprocessing, and writing output files, where these jobs take a little portion of the total runtime. The iterative bridging stage only takes $\sim 1\%$ of the total runtime. The major consumers of runtime are the module placement and the dual-defect net routing.

The module placement averagely consumes 66.89% of the total runtime on all the benchmarks. Since the solution quality of module placement can significantly affect the following routing stage, we run more iterations of the SA process to ensure that the total estimated wirelength and the volume are both optimized well. If the number of iterations is not enough for the optimization of the total wirelength and the total volume, we may not even obtain a legal solution in the routing stage due to heavy routing congestion. In our implementation, we ran 2000–3000 iterations for each benchmark.

The dual-defect net routing averagely consumes 31.89% of the total runtime on all the benchmarks. In most cases, the runtime of routing stage is less than the placement stage. Because a good module placement with less wirelength can facilitate the routing process, we put much more effort on the placement stage. Moreover, 85%–95% nets can be successfully routed in the first routing iteration, which takes $\sim 70\%$ of the total routing runtime in most cases. Finally, the rip-up and reroute process takes $\sim 30\%$ of the total routing runtime.

4) Visualization: Figure 20(a) and Figure 20(b) show the layouts of 4gt10-v1_81 and 4gt4-v0_73 respectively. We can see that all modules are compact with a small space-time volume.

V. CONCLUSION

In this paper, we have presented an effective algorithm that optimizes space-time volumes of TQEC circuits while considering time-ordered measurement constraints and the integration of state distillation boxes. We have proposed an iterative bridging technique that efficiently constructs bridge structures for dual-defect loops. Under the placement-and-routing framework, we have developed a time-ordering-aware 2.5D placement method that simultaneously optimizes the circuit volume and meets the time-ordered measurement constraints, and we have also proposed the friend net-aware dual-defect net routing that can improve the routability under topological deformation. Experimental results have shown that our proposed algorithm can compress much more space-time volume of TQEC circuits than state-of-the-art work.

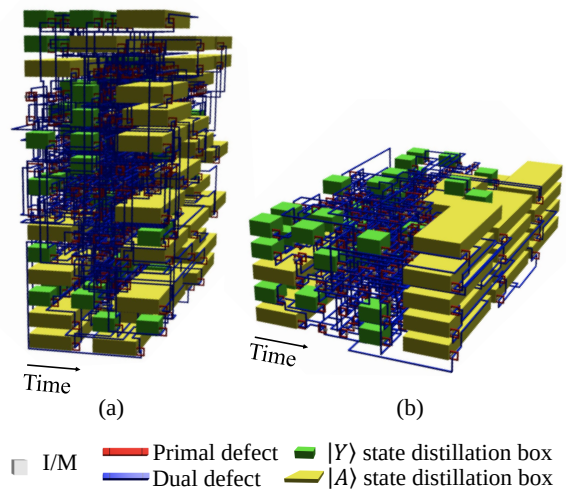

Fig. 20. Visualization. (a) The layout of 4gt10-v1_81. (b) The layout of 4gt4-v0_73.

REFERENCES

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of ACM Symposium on Theory of Computing*, Philadelphia, PA, July 1996, pp. 212–219.

[3] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, no. 3, p. 032324, September 2012.

[4] A. Paler, A. G. Fowler, and R. Wille, "Reliable quantum circuits have defects," *XRDS: Crossroads, The ACM Magazine for Students*, vol. 23, no. 1, pp. 34–38, September 2016.

[5] R. Raussendorf, J. Harrington, and K. Goyal, "Topological fault-tolerance in cluster state quantum computation," *New Journal of Physics*, vol. 9, no. 6, p. 199, June 2007.

[6] A. G. Fowler and K. Goyal, "Topological cluster state quantum computing," *arXiv preprint arXiv:0805.3202*, February 2009.

[7] A. Paler, S. J. Devitt, and A. G. Fowler, "Synthesis of arbitrary quantum circuits to topological assembly," *Scientific Reports*, vol. 6, no. 1, pp. 1–16, August 2016.

[8] A. Paler, I. Polian, K. Nemoto, and S. J. Devitt, "A fully fault-tolerant representation of quantum circuits," in *International Conference on Reversible Computation*, Grenoble, France, June 2015, pp. 139–154.

[9] ——, "Fault-tolerant, high-level quantum circuits: form, compilation and description," *Quantum Science and Technology*, vol. 2, no. 2, p. 025003, April 2017.

[10] S. Devitt and K. Nemoto, "Programming a topological quantum computer," in *Proceedings of IEEE Asian Test Symposium*, Niigata, Japan, November 2012, pp. 55–60.

[11] A. Paler, S. Devitt, K. Nemoto, and I. Polian, "Synthesis of topological quantum circuits," in *Proceedings of IEEE/ACM International Symposium on Nanoscale Architectures*, Amsterdam, Netherlands, July 2012, pp. 181–187.

[12] A. Paler, S. J. Devitt, K. Nemoto, and I. Polian, "Mapping of topological quantum circuits to physical hardware," *Scientific Reports*, vol. 4, no. 1, pp. 1–10, April 2014.

[13] C.-C. Lin, S. Sur-Kolay, and N. K. Jha, "PAQCS: Physical design-aware fault-tolerant quantum circuit synthesis," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 23, no. 7, pp. 1221–1234, July 2015.

[14] A. Paler, A. G. Fowler, and R. Wille, "Synthesis of arbitrary quantum circuits to topological assembly: Systematic, online and compact," *Scientific Reports*, vol. 7, no. 1, pp. 1–16, September 2017.

[15] A. Paler, "SurfBraid: A concept tool for preparing and resource estimating quantum circuits protected by the surface code," *arXiv preprint arXiv:1902.02417*, February 2019.

[16] M. AlFailakawi, I. Ahmad, L. AlTerkawi, and S. Hamdan, "Depth optimization for topological quantum circuits," *Quantum Information Processing*, vol. 14, no. 2, pp. 447–463, February 2015.

[17] A. Paler, R. Wille, and S. J. Devitt, "Wire recycling for quantum circuit optimization," *Physical Review A*, vol. 94, no. 4, p. 042337, October 2016.

[18] N. A. B. Adnan and S. Yamashita, "Logical qubit layout problem for ICM representation," *Journal of Information Processing*, vol. 26, pp. 20–28, January 2018.

[19] S.-i. Minato, "πDD: A new decision diagram for efficient problem solving in permutation space," in *International Conference on Theory and Applications of Satisfiability Testing*, Ann Arbor, MI, June 2011, pp. 90–104.

[20] A. G. Fowler and S. J. Devitt, "A bridge to lower overhead quantum computation," *arXiv preprint arXiv:1209.0510*, April 2012.

[21] A. Paetznick and A. G. Fowler, "Quantum circuit optimization by topological compaction in the surface code," *arXiv preprint arXiv:1304.2807*, April 2013.

[22] Y. Lin, B. Yu, M. Li, and D. Z. Pan, "Layout synthesis for topological quantum circuits with 1-D and 2-D architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1574–1587, August 2018.

[23] A. G. Fowler and G. Craig, "Low overhead quantum computation using lattice surgery." *arXiv preprint arXiv:1808.06709*, 2018.

[24] J.-A. Ali, G. Pranav, H. Adam, F. Diana, B. Kenneth, M. Margaret, and C. Frederic, "Optimized surface code communication in superconducting quantum computers." in *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture*, Massachusetts, USA, 2017, pp. 692–705.

[25] D. Herr, F. Nori, and S. J. Devitt, "Optimization of lattice surgery is NP-hard," *Npj Quantum Information*, vol. 3, no. 1, pp. 1–5, 2017.

[26] K. Asai and S. Yamashita, "Compaction of topological quantum circuits by modularization," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 102, no. 4, pp. 624–632, April 2019.

[27] S. Bravyi and A. Kitaev, "Universal quantum computation with ideal Clifford gates and noisy ancillas," *Physical Review A*, vol. 71, no. 2, p. 022316, February 2005.

[28] M. A. Nielsen and I. Chuang, *Quantum computation and quantum information*. Cambridge University Press, 2001.

[29] P. Falkenstern, Y. Xie, Y.-W. Chang, and Y. Wang, "Three-dimensional integrated circuits (3D IC) floorplan and power/ground network co-synthesis," in *Proceedings of IEEE/ACM Asia and South Pacific Design Automation Conference*, Taipei, Taiwan, January 2010, pp. 169–174.

[30] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and W. Shu-Wei, "B*-trees: A new representation for non-slicing floorplans," in *Proceedings of ACM/IEEE Design Automation Conference*, Los Angeles, CA, June 2000, pp. 458–463.

[31] L. McMurchie and C. Ebeling, "PathFinder: A negotiation-based performance-driven router for FPGAs," in *Proceedings of ACM International Symposium on Field-Programmable Gate Arrays*, Monterey, CA, February 1995, pp. 111–117.

[32] W.-H. Liu and Y.-L. Li, "Negotiation-based layer assignment for via count and via overflow minimization," in *Proceedings of IEEE/ACM Asia and South Pacific Design Automation Conference*, Yokohama, Japan, January 2011, pp. 539–544.

[33] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao, "NCTU-GR 2.0: Multithreaded collision-aware global routing with bounded-length maze routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 5, pp. 709–722, May 2013.

[34] M.-P. Wong, W.-H. Liu, and T.-C. Wang, "Negotiation-based track assignment considering local nets," in *Proceedings of IEEE/ACM Asia and South Pacific Design Automation Conference*, Macau, China, January 2016, pp. 378–383.

[35] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, Boston, MA, June 1984, pp. 47–57.

[36] C.-H. Hsu, W.-H. Lin, W.-H. Tseng, and Y.-W. Chang, "A bridge-based compression algorithm for topological quantum circuits." in *Proceedings of ACM/IEEE Design Automation Conference*, Taipei, Taiwan, December 2021.

[37] The LEMON graph library. [Online]. Available: https://lemon.cs.elte.hu/trac/lemon

[38] The Boost C++ libraries. [Online]. Available: https://www.boost.org/users/history/version_1_72_0

[39] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: An online resource for reversible functions and reversible circuits," in *International Symposium on Multiple Valued Logic*, Dallas, TX, May 2008, pp. 220–225.